Intention-Sensing Recipe Guidance via User Accessing to Objects

AUTHOR(S):

Hashimoto, Atsushi; Inoue, Jin; Funatomi, Takuya; Minoh, Michihiko

Intention-Sensing Recipe Guidance via User Accessing Objects

Atsushi HASHIMOTO*[1], Jin INOUE[1], Takuya FUNATOMI[2], and Michihiko MINOH[1]

[1]Kyoto University
[2]Nara Institute of Science and Technology,

May 17, 2016

Sensing the intention of a user's forthcoming action is a necessary function for systems that assist human physical activity. In this paper, we investigate a strategy for recipe guidance systems that can predict the forthcoming intended sub-task in a cooking task. We focus on user accessing objects, i.e., touching and releasing objects. Touching can indicate the start of the forthcoming sub-task and releasing can indicate the end of the task. The main difficulty lies in the fact that humans may move objects because they are in the way and use cooking tools that are unanticipated by an assistive system. In such cases, the accessed object should not indicate the forthcoming sub-task. Our contribution is that we propose a method to track the progress of a task based on the object access history. This enables us to eliminate object accesses that are out of context. Simultaneously, the method predicts the forthcoming sub-task based on a combination of progress and materials rather than tools and materials. We develop a guidance system that runs as a web service. In experiments, we observe real cooking activities navigated by this system. The Wizard of OZ method is utilized to simulate a system that detects object accesses. The experimental results show that we achieve 73.6% accuracy in the selection of the displayed information. This result supports the use of "access to objects" realize effective intention-sensing systems.

1 Introduction

Recent progress in mobile and multi sensing technologies has increased the opportunity for computers to assist human physical activity in the real world. Differing from human–computer interaction while doing desk work, people performing physical tasks concentrate on the task rather than constantly looking at a computer interface. "Intention sensing" is a necessary function in the context of assisting physical tasks.

For example, in exosuits developed to aid nursing care, myoelectric signals play an important role in sensing a wearer's forthcoming intented motion. Such signals can

*ahasimoto@mm.media.kyoto-u.ac.jp

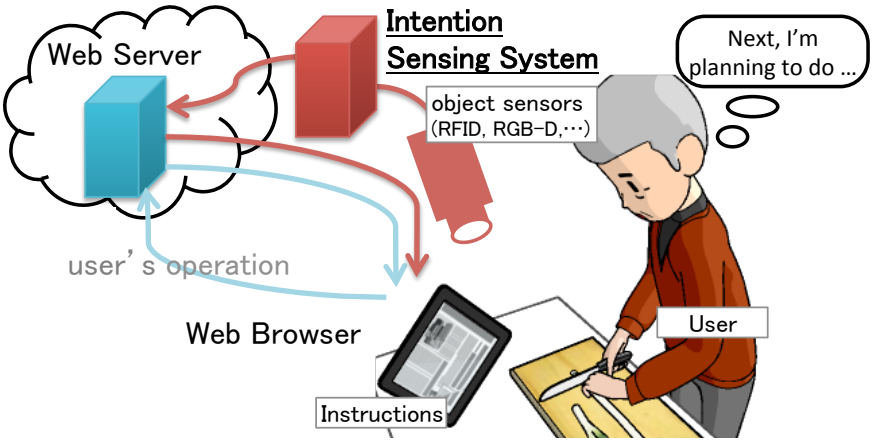

Figure 1: Assistive system with intention-sensing system.

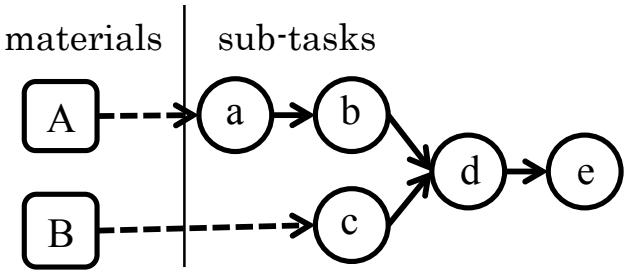

Figure 2: Workflow graph representation of a recipe.

automatically control motors without interface operations; thus, users can concentrate on nursing care tasks.

We aim to achieve an intention-sensing function for relatively more complicated tasks such as cooking (Fig. 1). A cooking task, such as making a salad, comprises many sub-tasks, e.g., cutting onions, tearing lettuce, mixing ingredients, and dressing the mixture. The order in which the sub-tasks are executed may not be strictly defined (Hamada, Okabe, Ide, Sakai, & Tanaka, 2005), and can be described as a workflow in which one sub-task corresponds to one node (Fig. 2). Because workflow description is often used for many kinds of physical tasks (e.g. assembling tasks), a recipe guidance scenario can be applied to these tasks. In this context, recipe guidance is favorable because small mistakes in cooking tasks are rarely critical. This encourages more natural human behavior than that in a highly disciplined task. In this sense, a cooking task is appropriate for achieving intention sensing in a practical situation.

Intention sensing in a cooking context can be defined as the estimation of the next sub-task in the workflow graph. To solve this problem, we use "object access," i.e., touching and releasing objects, as a substitute for myoelectric signals. The worker

naturally accesses various objects while cooking. Accessed objects are good indicators for predicting the forthcoming sub-task. Predicting the next sub-task enables us to help a user by displaying multimedia information about what the user will do next. This will free the user from frequent interface operations that interfere with user concentration.

On the other hand, differing from myoelectric signals, an object access does not always correspond to user intention. A user may grab an object simply to set it aside. The user may also use an unanticipated tool to execute a sub-task. We refer to the former problem as "deceptive access," and latter as "unanticipated tool." Given these problems, predicting the forthcoming sub-task in a practical situation is challenging.

The primary contribution of this paper is an algorithm that deals with the above problems by tracking the progress of a task according to the object access history. Most deceptive accesses are expected to be out of context, and progress-tracking provides context information to eliminate such out-of-context accesses. Simultaneously, we combine the progress information with the accessed materials. Even when the tools used in a sub-task are unanticipated, relative to the known progress, the accessed materials provide sufficient information to identify the forthcoming sub-task.

2 Related work

Promoting better eating habits is important for people's health and well-being and for reducing health care costs. To guide people toward better eating habits, it is important to assist them in preparing healthy, delicious, and economical dishes. In recent years, many such recipes have become available on the World Wide Web. Video and still images facilitate user understanding of various cooking techniques, and voiced narration helps users not to miss necessary steps. However, such multimedia content has a trade-off problem, i.e., the more multimedia contents a recipe contains, the more operations are required to search and play the content.

To improve accessibility to such multimedia content, many researchers have proposed various interfaces (Ju, Hurwitz, Judd, & Lee, 2001; Bradbury, Shell, & Knowles, 2003; Hamada et al., 2005; Nintendo, 2006; Miyawaki & Sano, 2008; Uriu et al., 2012; Matsushima, Funabiki, Zhang, Nakanishi, & Watanabe, 2013). Similar interfaces have also been proposed for assembly tasks (Zauner, Haller, Brandl, & Hartmann, 2003; Tang, Owen, Biocca, & Mou, 2003; Yuan, Ong, & Nee, 2008; Siltanen et al., 2007). The main difference between cooking and assembly tasks is the availability of radio-frequency identification (RFID) tags and visual markers, i.e., they are unavailable for foodstuffs. Nonetheless, we should consider assembly tasks because they can also be represented by a workflow graph, such as that shown in Fig. 2.

Many systems were developed in the early 2000s (Ju et al., 2001; Nintendo, 2006; Zauner et al., 2003; Tang et al., 2003). Those systems recognize the progress of a cooking/assembly task by tags, markers, or simple user operations. However, the recipes in those systems are described as fully-ordered sub-tasks, and the workers must follow the described order strictly. In such a situation, the next sub-task is always defined uniquely, and there is no room to arrange the scenario according to user intention.

Hamada et al. (Hamada et al., 2005) pointed out that the flexibility of a cooking recipe can be represented by a workflow graph (Fig. 2). Their system is intended to

assist users in preparing a hot meal. For this purpose, the system recommends the complete-order of sub-tasks by an online scheduling algorithm such that all dishes comprising the meal are completed at the same time. A partial order defined by the workflow graph is an important condition for the online scheduling algorithm. This work focused more on the scheduling algorithm, and less on the interface. This system helps users a great deal, particularly when they are unfamiliar with cooking procedures. On the other hand, the absence of intention-sensing functionality will increase user burden whenever the user does not follow the plan recommended by the system. For an experienced cook, this will decrease their motivation to use the system even though the scheduling is a helpful function. Our purpose is to maintain user initiative by sensing their intention while providing various types of computer-based assistance, including scheduling assistance.

Some studies have considered user intention. For example, Yuan developed an interface that enables users to signal the end of sub-task execution and select the next sub-task independently (Yuan et al., 2008). Similarly, some cooking assistance systems use special devices to access information freely while executing a task (Bradbury et al., 2003; Matsushima et al., 2013). Clearly, interface operation can interrupt work frequently and thereby interfere with user concentration.

Rather than explicit interface tools, Miyawaki proposed a system that uses real objects as implicit interfaces for selecting sub-tasks (Miyawaki & Sano, 2008). Their system achieved automatic recipe navigation without explicit interfaces. To achieve full automation, they preliminarily form a correspondence between the sub-tasks and the objects on the cooking counter, which are mainly containers. The system identifies the forthcoming sub-task according to the container that the user accessed. This approach is similar to ours; however, there is an essential difference. Miyawaki et al. avoided the ambiguity for the forthcoming sub-task by assigning unique objects to each individual sub-task. Under normal circumstances, unique correspondences between sub-tasks and objects are not guaranteed. The user will easily forget such unnatural correspondences, and this approach will fail when the user uses an unanticipated object. The proposed method relies on the correspondence between sub-tasks and objects; however, the correspondence is determined by the nature of each sub-task, which users do not need to memorize. In this setting, the objects assigned to sub-tasks are no longer unique. Rather than relying on unique correspondences, we propose an algorithm that focuses on identifying the intended sub-task from contextual information and the combination of accessed objects.

Schneider simulated human execution of a cooking task using a dynamic Bayesian network (DBN) (Schneider, 2009). In the DBN, a state corresponds to the cooking progress, and a transition corresponds to a sub-task that the user has executed. In this problem setting, the system attempts to determine the recipe that the user is working with rather than what sub-task the user will perform next. Though the system originally attempts to identify recipe, the process of identification on a DBN includes progress-tracking on a completely-ordered sub-tasks.

However, this method can not track cooking progress when a recipe is described in partial-order. In their work, the DBN, which represent a recipe, was generated randomly and no human factors were considered. Moreover, the number of possible states of the DBN increases combinatorially if the system allows sub-tasks to be executed

flexibly (Appendix B). We deal with the flexibility of a recipe and focus more on the actual cooking activity. The heuristics of our algorithm reveal the sparseness of human state transition behavior even with a combinatorially explosive number of states.

3 Sensing intention via object access

3.1 Workflow graph representation of a recipe

Before discussing our approach in detail, we define the workflow representation of the graph more strictly. Let $G(V, E)$ be the workflow graph shown in Fig. 2, where $v \in V$ corresponds to a sub-task, and $e \in E$ defines the order relationship between two sub-tasks. The edge is given in the following manner. We express an order relationship between two sub-tasks v_1 and v_2 by the operator $<$, i.e., $v_1 < v_2$ means that v_1 must be completed before starting v_2. To avoid redundant edges to represent the partial order definition in G, we add a directed edge $\{v_1, v_2\}$ to E if and only if $v_1 < v_2$ and there are no other sub-tasks v such that $v_1 < v$ and $v < v_2$.

For example, the graph shown in Fig. 2 consists of subtasks $V = \{a, b, c, d, e\}$, and order definitions $E = \{\{a, b\}, \{b, d\}, \{c, d\}, \{d, e\}\}$. The graph has two paths $\{a < b < d < e\}$ and $\{c < d < e\}$. Each path of sub-tasks in the flow tracks one material, and each conjunction of paths corresponds to a material mixing sub-task. As G is intended to represent the partial order relationship of a task, there are no order definitions between sub-tasks $\{a, b\}$ and c. In other words, the users can arrange the order of these sub-tasks.

To utilize object access to sense user intention, we associate object labels with each sub-task in V. Let O_v be the labels linked to sub-task v. We also maintain the division between materials and tools by denoting them O_v^m and O_v^t ($O_v^m \cup O_v^t = O_v$ and $O_v^m \cap O_v^t = \phi$,) respectively. In a cooking activity, whether a seasoning belongs to O_v^m or O_v^t is not obvious. To rule out this ambiguity, we define that an object is a material if and only if it is processed alone in any of v; otherwise it is a tool. For example, oil is a material if G has a sub-task v in which the oil is heated by itself, but it is a tool when it is always added to other materials. This definition avoids the appearance of v with no materials in linked object labels.

Given the nature of the workflow graph, materials are further categorized into single material and mixtures of materials. The tool and single material labels can be preassigned to v manually. Such remain unchanged until the materials are mixed. For example, we set $O_d^m = \{A, B\}$ at sub-task d in Fig. 2 because A and B are successively involved in the process at sub-task d. After a sub-task that mixes two or more materials, the labels of the mixed materials are combined as a single label that indicates that indicates a mixture of materials. In Fig. 2, after the process at sub-task d, A and B are mixed and treated as a material represented by $\{A, B\}$, and we set $O_e^m = \{\{A, B\}\}$. Thus, O_v^m is obtained automatically from the assigned raw materials A at sub-task a, B at sub-task c, and the structure of G. A more precise definition of the algorithm to assign O_v^m for each sub-task is given in Appendix A

A recipe can be represented by the various granularities of the sub-tasks V. For example, a recipe can represent the direction for tomato puree by a single sub-task that says “puree tomato,” or several sub-tasks, such as “remove tometo stems,” “blanch the

tometoes," "put blanched tomatoes into a pot of cold water," "remove the skin," "dice the tomatoes," and "puree diced tomatoes in a food processor." Note that the granularity affects the usability of the system. Generally, the level of detail is up to the author of the given recipe. For beginners, more detail is preferred; for experienced cooks, a less detailed explanation may be better.

Basically, the more finely the sub-tasks are divided, the more detailed information the system can display. A heighly granular division may be helpful for users who are unfamiliar with the recipe. For those who are familiar with the recipe, they are not annoyed by too much information when the system senses their intention and the displayed information is switched automatically. Because our goal is a system with intention sensing, the system should be able to provide finely divided sub-tasks that help users with unfamiliar recipes and does not annoy them with too many sub-tasks for users who are familiar with the recipes. To make the recipes more compatible with our object access strategy, we divide a recipe into sub-tasks such that each sub-task has the smallest O_v. O_v is smallest when a sub-task does not include processes that use different tools. Thus, the smallest O_v is obtained easily by dividing sub-tasks that include two processes that use different tools into separate sub-tasks.

3.2 Basic problems with sensing intention from object access

We use object access to achieve intention-sensing functionality. Let O^h be a set of objects held in hand. Intention sensing via object access is basically considered a problem of finding v intended as the next sub-task based on the comparison of O^h with O_v for all $v \in V$. However, there are some practical problems that should be considered. First, when sub-tasks u and v have the same object labels ($O_u = O_v$) and are matched to O_h, we must consider any external evidences that indicates whether u or v is more likely to be performed next. To address this problem, we select the sub-task that is on the same path as the previously performed sub-task. Second, O^h and O_v are not always reliable due to deceptive access and the use of unanticipated tools.

Deceptive access is object access that is not related to the user's intention. Typically, it occurs when the user moves objects without the intention of using them. This type of action is common in many types of tasks. Such actions make a naive rule-based system, such as that proposed in (Miyawaki & Sano, 2008), work improperly.

Unanticipated tools must be considered because it is obviously impossible to list all possible tools for v in advance. When an unanticipated tool is used as a substitute tool in O_v, O_v and O_h become different. Thus, the system must accept a certain difference of tools in O_v and O_h.

To account for deceptive access, we maintain the progress of the task on G and reject any access that does not suit to the situation. To handle the unanticipated tools, we have designed a score function that can deal with a certain amount of mismatch between O_v and the actually accessed objects.

3.3 Enhancing robustness against deceptive access

To eliminate deceptive accesses from the intention-sensing process, it is helpful to track the progress on a workflow graph $G(V, E)$. Here, progress is described by a set of

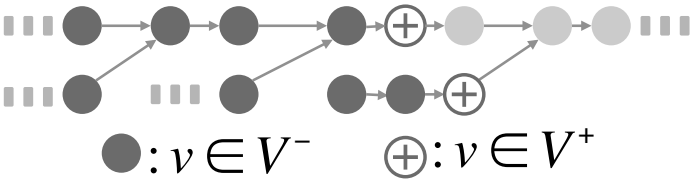

Figure 3: Context information V^- and V^+ on a workflow graph.

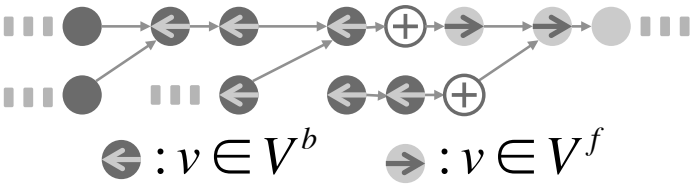

Figure 4: Expanded context of V^b and V^f.

completed sub-tasks $V^-(\subseteq V)$. V^- is obtained by detecting every completion of $v \in V$ through observation. Sub-tasks that are ready for execution are obtained as sub-tasks whose ancestors are all in V^-, where u is an ancestor of v if $u < v$. We refer to such sub-tasks as V^+ (Fig. 3).

A simple idea for eliminating deceptive accesses is to ignore an access if the accessed object is not listed in $\cup_{v \in V^+} O_v$. This strategy will eliminate deceptive accesses effectively; however, in practice, it is too naive against inaccurate V^+. When a single mistake occurrs in completion detection, V^- becomes inaccurate and consequently V^+ will also be affected.

We consider a situation where the actual value of V^- differs from the value in the system. Consider a set of sub-tasks $\hat{V}^-$ that have been estimated as complete by the system and distinguish this set from V^-, which is an accurate set of completed sub-tasks. Similarly, $\hat{V}^+$ is the set induced by $\hat{V}^-$. Note that, a sub-task that is in V^+ and on a path in G is always unique. We refer to $\hat{v}$ as a subtask that is in $\hat{V}^+$ and on the same path as $v \in V^+$. When sub-tasks $\hat{v}$ and v are differ, accesses to objects in O_v are ignored and the system fails to display instructions for v.

To improve robustness against failure in $\hat{V}^-$, we extend $\hat{V}^+$ in two directions and obtain modified contexts $\hat{V}^f$ and $\hat{V}^b$, where $\hat{V}^f$ consists of nodes in the forward direction of directed edges in G from those in $\hat{V}^+$, and $\hat{V}^b$ consists of nodes in the backward direction (Fig. 4). $\hat{V}^b \cup \hat{V}^+ \cup \hat{V}^f$ will cover more nodes in V^+, which is the accurate context, than $\hat{V}^+$. Therefore, involving accesses related to $\hat{V}^f$ and $\hat{V}^b$ in addition to those related to V^+ will reduce the risk of eliminating access to $v \in V^+$.

It is important to consider how to expand $\hat{V}^+$. For example, a wider context results in greater loss of contextual information. On the other hand, narrow expansion leads to the same result obtained when using V^+. To decide our strategy, we focused on the difference of risks between two types of failure on $\hat{v} \in \hat{V}^-$, i.e., $\hat{v} < v$ and $v < \hat{v}$, which we refer to as *left-unattended* error and *overrunning* error, respectively.

When $\hat{v} \in \hat{V}^+$ satisfies in the condition $\hat{v} < v$, an access to objects in $O_{\hat{v}}$ is necessary to recover. However, objects in $O_{\hat{v}}$ are hardly accessed in this case because $\hat{v}$ is already completed. Hence it is hard to recover automatically from the left-unattended error. In contrast, when $v < \hat{v}$, the user proceeds the task and, in the meanwhile, the user will access to objects in $O_{\hat{v}}$ to work on $\hat{v}$. Hence, the system can potentially recover from the overrunning error.

To avoid the left-unattended error with certainty, we employ a completion detector for $\hat{V}^-$ such that it achieves a high recall rate, thereby hardly missing the completion. This strategy leads to many overrunning errors. To reduce the effect of overrunning errors, we expand the context $\hat{V}^+$ in only one sub-task ($\hat{V}^f$). In addition, we expand the context broadly in the backward direction such that v tends to lie in $\hat{V}^b$ while $v < \hat{v}$ (Fig. 4).

Here, we provide a mathematically rigorous description of the expansion shown in Fig. 4. First, $\hat{V}^f$ is obtained as $\{v|\{u, v\} \in E \wedge u \in \hat{V}^+\}$. Second, we add u to $\hat{V}^b$ if and only if $u < v$ for any $v \in \hat{V}^+$ and there is at most one conjunction on the path from u to v. This backward expansion accepts accesses to both materials mixed at the conjunction sub-task and the mixture of materials produced by the sub-task. Adding $\hat{V}^b$ and $\hat{V}^f$ obtained in this manner to $\hat{V}^+$ loses some contextual restriction but is more robust against corruption in $\hat{V}^-$, which may contain errors.

3.4 Dealing with unforeseen tools

Differing from the system proposed in (Miyawaki & Sano, 2008), our system does not force users to obey any one-to-one correspondences between tools and sub-tasks. Instead, we list as many alternative tools as possible in O_v and calculate a matching score between the objects held in hand and in O_v for each v. Since alternative tools are not used together, we prepare O_v for each alternative tool, i.e., $O_v = \{O_v^i | 0 \leq i < N_v\}$, where N_v is the number of alternative tool combinations. For example, if a sub-task is "peel the potato," the tool can be a knife or a peeler. In that case, O_v consists of $O_v^0 = \{''knife'','' potato''\}$ and $O_v^1 = \{''peeler'','' potato''\}$.

When any O_v^i is equal to O^h, it is easy to sense the intention; however, as discussed above, O^h and O_v^i are frequently not equal. The user may execute sub-task v with an unforeseen tool, or keep holding unnecessary tools during a sub-task. In such cases, O^h differs from any of O_v^i.

To match O^h to O_v^i while allowing such difference in tools, we employ a heuristic scoring function $L(v; O)$, and select the sub-task with the highest score. The score for v is calculated as follows;

$$L(v; O^h) = \begin{cases} 0 & v \notin \hat{V}^b \cup \hat{V}^+ \cup \hat{V}^f \\ \max_i L(O_v^i; O^h) & otherwise \end{cases} \quad (1)$$

$$L(O_v^i; O^h) = \sum_{o \in O_v^i} \sigma(o) \mathbb{1}(o \in O^h) - \varepsilon |O^h \backslash O_v^i|, \quad (2)$$

where $\sigma(o)$ is an elemental score assigned to each object in O_v^i. We assign a higher score to materials in O_v^i and a smaller score to tools because the material is a dominant

Table 1: Elemental scores and penalty.

$\sigma(o)$	materials (with tools)	0.75
	tools (with materials)	0.25 / # of tools in O_v^i
	materials (w/o tools)	1.0
	tools (w/o materials)	0.95 / # of tools in O_v^i
ε	0.04	

factor in intention sensing. $\mathbb{1}(o \in O)$ is an indicator function that returns 1 if $o \in O$ is true; otherwise, it returns 0. ε is a penalty score for extra objects, and $|O|$ is the number of elements in set O.

We show the settings fors elemental score $\sigma(o)$ and penalty score ε in Table 1. These values were obtained in the following heuristic manner. Essentially, the score for tools should only be used to differentiate sub-tasks that treat the same material. An exception is a case wherein only tools are accessed. A typical example is seasonings, which are often sprinkled on foodstuffs, and the foodstuffs themselves are not accessed. The seasonings, which we consider tools, will be the dominant factor in such cases. Thus, we give $\sigma(o)$ a high score for such tools if O_v^i contains no materials.

3.5 Overall algorithm

Here, we present the details of our algorithm. The overall algorithm is given in Algorithm 1. In the algorithm, v^c is the sub-task whose instructional multimedia content is displayed to the user.

$\mathcal{R}$ and C are external functions. $\mathcal{R}$ is a recommendation function that chooses a sub-task in the order that the system believes to be the best without considering user intention. One example of $\mathcal{R}$ is the scheduling algorithm proposed in (Hamada et al., 2005). Not that the design of $\mathcal{R}$ is not the focus of this paper; therefore, we simply use a static, preliminary determined order as the return value of $\mathcal{R}$. $\mathcal{R}$ is only called when our algorithm cannot estimate the forthcoming intended sub-task (lines 2 and 24). $C(v)$ detects the completion of $\hat{V}^-$. This function is called whenever an object is released. The details of these functions are given in Section 3.6.

The procedure in line 5 was given in Section 3.3. The procedure from lines 7 to 14 maintains O^h according to the detection of each object access. Lines 15 to 24 estimate the intended sub-task. Since $\hat{V}^b$ is primarily used for recovery, we give priority to $\hat{V}^+ \cup \hat{V}^f$. When several sub-tasks demonstrate the highest score, we adopt the sub-task that appears first in the order recommended by $\mathcal{R}$.

3.6 Design of completion detector $C(v)$

$\hat{V}^-$ is maintained by detecting the completion of sub-tasks whenever the user releases objects. There are many action recognizers, some of which are specialized for culinary tasks (Rohrbach, Amin, Andriluka, & Schiele, 2012; Packer, Saenko, & Koller, 2012; Lei, Ren, & Fox, 2012; Iscen & Duygulu, 2013; Shimada, Kondo, Deguchi, Morin, & Stern, 2013; Kuehne, Fraunhofer, Arslan, & Serre, 2014). These methods can be a

Algorithm 1 Complete forecast process.

1: $O^h = \phi$, $\hat{V}^+ = \{v | v \in$ source vertices in $G\}$, $\hat{V}^- = \phi$
2: $v^c = \mathcal{R}(\hat{V}^+, \phi)$
3: **while** $\hat{V}^- \neq V$ **do**
4: display v^c
5: maintain $\hat{V}^b$, $\hat{V}^+$, and $\hat{V}^f$ along with $\hat{V}^-$
6: wait until any object o is touched or released
7: **if** o is newly touched **then**
8: $O^h = O^h \cup \{o\}$
9: **else if** o is newly released **then**
10: $O^h = O^h \backslash \{o\}$
11: **if** $C(v^c)$ is true **then**
12: $\hat{V}^- = \hat{V}^- \cup v^c$
13: **end if**
14: **end if**
15: $v^+ = \underset{v \in \hat{V}^+ \cup \hat{V}^f}{\text{argmax}} L(v; O^h)$
16: $v^- = \underset{v \in \hat{V}^b}{\text{argmax}} L(v; O^h)$
17: **if** $L(v^+; O^h) \geq 1.0$ **then**
18: $v^c = v^+$
19: **else if** $L(v^-; O^h) \geq 1.0$ **then**
20: $v^c = v^-$
21: **else if** $L(\cdot +) > 0.0$ **then**
22: $v^c = v^+$
23: **else**
24: $v^c = \mathcal{R}(\hat{V}^+, v^c)$
25: **end if**
26: **end while**

good solution for the completion detection problem. The purpose of this paper is not the development of an accurate completion detector; therefore, rather than implementing those action recognizers, we have implemented a simple detector based on only two indicators, i.e., the time elapsed since v^c was displayed and the confidence of the displayed information $L(v^c; O^h)$, both of which are obtained by executing Algorithm 1.

Let n be the iteration index of the loop in line 3 in Algorithm 1, and a subscript n, e.g. v^c_n, denotes the n-th displayed sub-task. In addition, t_n denotes the initial display time of v^c_n. Thus $t - t_n$ is the elapsed time at time t after v^c_n is displayed.

Note that C returns false if $t - t_n < \theta_t$ because all sub-tasks require a certain amount of time. We set $\theta_t = 1.0$ s for sub-tasks. We also disregard an object access for object o if o is released in 1.0 s. In this case, we reset all parameters in Algorithm 1 to those belonging to the state prior to object o being accessed. We then recalculate the procedure while ignoring the access to object o.

If the elapsed time is greater than 1.0 s, we check the second condition of the

highest record of $L(v; O^h)$ during term $[t_v, t]$ as follows:

$$C(v^c) = \begin{cases} true & L_{\max} \geq \theta_1 \\ true & \theta_1 > L_{\max} \geq \theta_2 \wedge \mathcal{R}(\hat{V}^+_{n-1}, v^c_{n-1}) = v^c_n \\ false & otherwise, \end{cases} \quad (3)$$

where $L_{\max} = \max_{[t_n,t]}(L(v^c; O^h))$. Here, θ_1 and θ_2 are higher and lower thresholds, respectively, obtained by the elemental scores in Table 1.

Let o^m_w, $o^m_{w/o}$, o^t_w, and $o^t_{w_o}$ be a material with tools, a material without tools, a tool with materials, and a tool without materials, respectively. Not that these correspond to the cases shown in Table 1. Then θ_1 is expressed by $\sigma(o^m_w) + \frac{\sum_{o^t_w} \sigma(o^t_w)}{2}$. Note that θ_2 must be less than $\sigma(o^m_w)$(we set it to 0.5).

C returns true if $L(v^c; O^h)$ becomes greater than θ_1. Even when $L(v^c; O^h)$ is less than θ_1, C returns true when the score is greater than θ_2 and v^c_n corresponds to $\mathcal{R}(\hat{V}^+_{n-1}, v^c_{n-1})$, where $\hat{V}^+_{n-1}$ is the value of $\hat{V}^+$ at the $(n-1)$-th iteration. This condition is derived from the idea that v^c_n should be executed consecutively after v^c_{n-1}. Note that C returns false for any other case.

At line 12, which updates $\hat{V}^+$, we push v^c and any sub-task v that satisfies $v < v^c \wedge v \notin \hat{V}^-$, into $\hat{V}^-$. Note that this completion-detecting algorithm can be combined with other sophisticated action recognizers. This remains a focus of future work.

4 Interface Implementation

4.1 Web-based platform CHIFFON

To achieve an interface that is controllable from the intention-sensing system, we implemented a recipe display system called CHef's Interface For Food preparatiON (CHIFFON). CHIFFON has server-side and client-side programs, and two-way communication is established between them (blue arrows in Fig. 5); client-to-server messages deliver all user operations to the system, and server-to-client messages are orders for clients rendering the instructional multimedia content in a web browser. Figure 6 shows the graphical user interface (GUI) rendered by the web browser.

The server-side program can receive a message from any external system, including the object sensor module shown in Fig. 1. When the server receives messages from an external system, it generates and sends a rendering order to the target client web browser (red arrows in Fig. 5).

We designed a recipe format that describes the workflow structure, text instructions, links to multimedia content, and O^i_v for each sub-task v. We also provided an application programming interface (API) for external systems.

The recipe format, called Hand-Work Markup Language (HWML), is an XML document whose structure and content pattern are defined in the format of RELAX NG schema format (Murata, 2001). This format allows users to describe the workflow structure of the sub-tasks. Each sub-task can have several triggers. A trigger is a set of conditions, and we describe O^i_v as a trigger in this study.

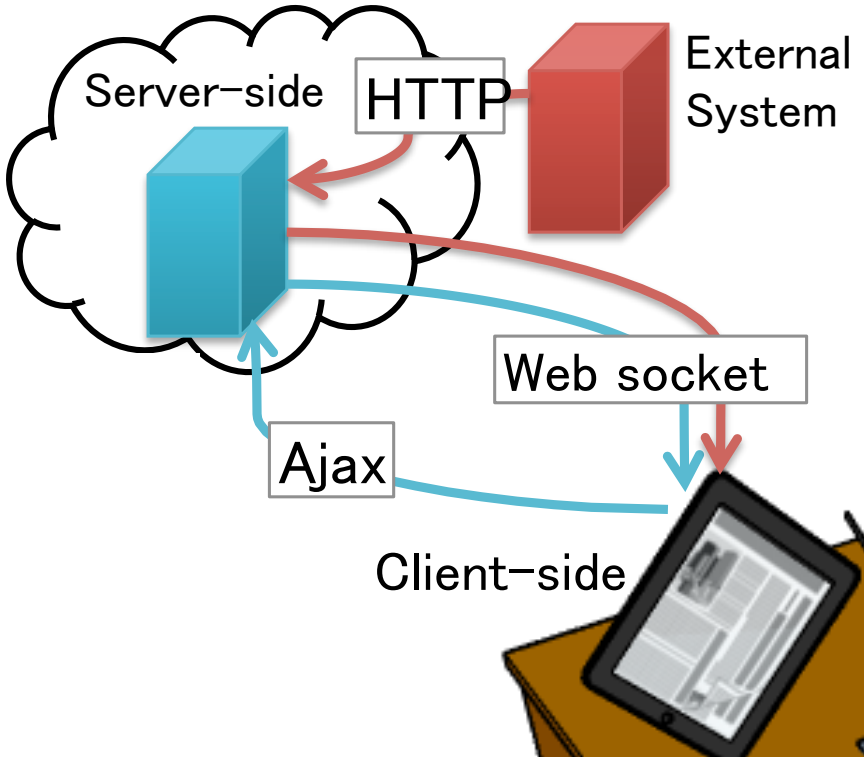

Figure 5: Communication protocols between modules.

The API receives various orders as an HTTP GET request as long as the request is written in the format specified by the API. Once the server-side program has accepted the order through the API, it sends a message to the client-side program. Therefore, external systems can control the client-side program in a sidelong manner through this API.

The intention-sensing algorithm proposed in the previous section is implemented in the server-side program. It was called when the object sensor module sends a message that notifies an object access through the API. Whenever the algorithm called, it calculates the most likely intended sub-task, and sends a rendering order to the target client web browser.

4.2 Object sensor implementation

Figuire 1 also shows object sensors. This module detects the user access to objects. An object access is described by a tuple $\{t, o, a(o)\}$, where t is a timestamp, o is an object label, and $a(o)$ is an object-access label, i.e., "touched" or "released." The output of this module is sent to the intention-sensing algorithm on the server via the CHIFFON API.

Figure 6: GUI of client-side program.

There are many choices to implement this module: RFID (Nakauchi, Fukuda, Noguchi, & Matsubara, 2005), RFID and load sensors (Chang et al., 2006), AR markers (Miyawaki & Sano, 2008; Ueda, Funatomi, Hashimoto, Watanabe, & Minoh, 2011), RGB-D cameras (Klompmaker, Nebe, & Fast, 2012), and multi-modal signal processing (Hashimoto et al., 2010, 2012; Yasuoka, Hashimoto, Funatomi, & Minoh, 2013). Each method has advantages and disadvantages, and there may be a best combination of methods to achieve an optimal solution.

Our focus is the design of an intention-sensing system, and developing such an optimal solution is beyond the scope of this study. Instead, we have adopted the Wizard of OZ method (WOZ) (Kelley, 1984) which has been widely accepted over the last 30 years for evaluating various types of intelligent interface prototypes (Sandweg, Hassenzahl, & Kuhn, 2000; Favela, Tentori, & Gonzalez, 2010; López, López, Guerrero, & Bravo, 2014). The WOZ method uses a human as a wizard who functions as a substitute for an intelligent module. In our case, the wizard provides an alternative to the object sensor via the interface shown in Fig. 7. The wizard stays in the background of the system so that subjects do not notice its presence. One significant advantage of this method is that interface designers do not have to wait for relevant technologies to mature sufficiently.

5 Experiments

There are two important factors that affect a system's proper intention sensing: non-unique O_v^i due to the complexity of the recipe and the nature of human activity discussed in Section 3.2. To evaluate each factor independently, we designed two experiments. In the first experiment, we restricted users to not having any deceptive accesses and using only the tools listed in O_v to compare the accuracy of the displayed information between two recipes with different complexity. In the second experiment, we removed these restrictions and evaluated the system with cooking activities in a natural setting.

Figure 7: Interface for wizards. When a button for object o is pushed, $a(o)$ = 'touched' is sent with a timestamp. Then, the button turns red. $a(o)$ = 'released' is sent after pushing the red button again.

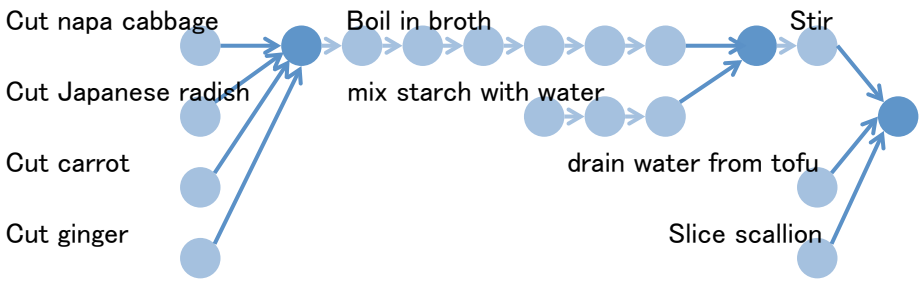

Figure 8: Overview of ginger soup. Nodes colored in dark blue are mixing sub-tasks.

5.1 Robustness against recipe complexity

Since several sub-tasks can potentially have the same combination of objects as O_v^i, the algorithm must be robust against such recipes. To evaluate robustness, we prepared two different recipes, i.e., "ginger soup" (19 sub-tasks) and "fried rice" (30 sub-tasks), as shown in Figs. 8 and 9, respectively

Ginger soup is a recipe that is easy to guide by object access because O_v^i for $^\forall v \in V$ is unique. In such a recipe, v can be identified uniquely without any contextual information. In contrast, fried rice has many sub-tasks that share the same set of objects as O_v^i. These sub-tasks are indicated by the circle border colors in Fig. 9. There are four pairs (orange, red, blue and green) and one triplet (pink) of sub-tasks that share the same O_v.

Prior to conducting the experiments, we gave the subjects the following restrictions.

1. Do not add any sub-tasks to the process; do only what the recipe instructs.
2. Do not touch any object unless it relates to what you intend to do.

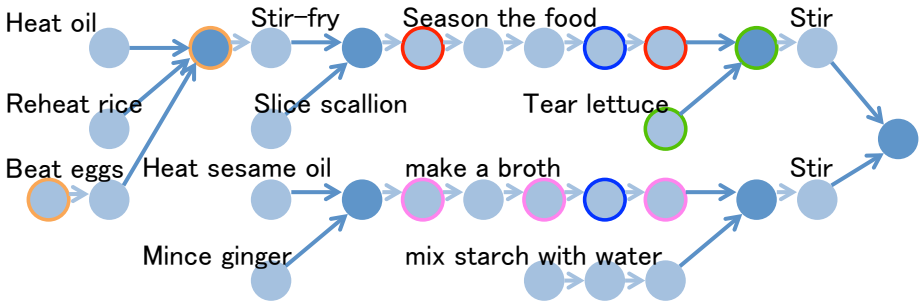

Figure 9: Overview of fried rice with starchy sauce. The same boundary colors indicate the duplicated O_v^i.

Table 2: Accuracy of displayed information and number of user operations.

	Ginger Soup		Fried Rice	
Subject	A	B	A	B
# of displayed correct sub-tasks	118	202	121	120
# of displayed false sub-tasks	16	18	7	6
Accuracy	88.1%	91.8%	94.5%	95.2%
# of user operations	0	2	0	0

3. Do not use a tool that is not called for in the recipe.

4. Do not release the objects while the sub-task is ongoing.

Two subjects were employed in this experiment, and both are skilled amateur cooks. It is too difficult to cook with the above restrictions without practice. Therefore, the subjects rehearsed several times.

Note that the infomation was displayed automatically during the rehearsals and the real part of the experiment. Simultaneously, the subjects were allowed to operate the interface manually on a touch display when they felt any inconvenience. The number of manual operations indicates the number of critical failures for the subjects. Thus, counting user operations enables us to count such failures separately from other failures that do not affect the interface's usability significantly.

Table 2 shows the accuracy of the displayed information and the number of user operations. Despite the duplicated O_v^i in the fried rice recipe, accuracy remained high for both recipes. From this result, we confirm that the algorithm works with both difficult and easy recipes.

It can also be said that the system was not operated frequently by the subjects in every case, even though the accuracy did not reach 100%. This is considered to be due to the difference in the timing of failure. During the experiments, we found that the subjects primarily concentrated on the cooking task and did not check the display all the time. It appears that most cases of displaying false sub-tasks occurred while the subjects concentrated on their task, which had little effect on the subjects.

Table 3: Number of manual operations while cooking fried rice recipe.

Subjects	A	B	C	D	E	G	H
Manual Operation (times)	2	2	9	1	2	5	3

Table 4: Accuracy of displaying steps and number of manual error corrections.

Subjects	A	B	C	D	E
a) expected access	99	101	72	98	77
b) deceptive access	92	106	73	76	95
c) # of displaying correct sub-tasks	128	162	108	135	121
d) # of displaying false sub-tasks	63	45	37	39	51
e) (d) at (a)	23	15	11	17	19
f) (d) at (b)	40	30	26	22	32
Accuracy: (c)/((c)+(d))	67.0%	78.3%	74.5%	77.6%	70.3%

5.2 Evaluation with natural human activities

To evaluate the system during natural human activity, we used the fried rice recipe, which is more challenging and practical than ginger soup recipe. Differing from the first experiment, we set no restrictions in this experiment. Seven subjects were recruited for this experiment (three males and four females). All subjects cook at least once a week, and thereby have a certain level of culinary skill. To enhance the naturalness of the activity, we gave the subjects 10 min to review the entire recipe before starting observation. In addition, we instructed them to make a plan for cooking the dish on an optimal schedule.

We also compensated for the unfamiliar cooking environment. The subjects did not know where things were stored; thus, it was expected that this would restrict user activity. In particular, cookware tends to be used more often when stored in an easy-to-find location. To cancel this effect, we asked the subjects to determine what kind of cookware they would need beforehand and placed it in plain view before starting the task.

The results of this experiment are shown in Tables 3 and 4. As can be seen in Table 3, subjects C and F were very sensitive to false display; they checked the display for each sub-task; and corrected errors nine and five times in total, respectively. It was revealed that our system is not very effective for this type of user. On the other hand, the remaining four subjects rarely operated the system. This indicates that object access can be a strong indicator to sense user intention.

We examined the accuracy of the displayed information for subjects A to E precisely. For all data, there were relatively more falsely displayed sub-tasks than the results shown in Table 2; however, subjects A, B, D, and E were not significantly confused by such falsely displayed sub-tasks. A typical error occurred at sub-tasks where materials are mixed, i.e., v^c went back and forth repeatedly between such a sub-task and its next sub-task (e.g., sub-tasks *d* and *e* in Fig. 2). This occurred in the following scenario. First, one of the target materials was touched, and the instruction for the mixing

sub-task was displayed. Then, the material was released, and the completion detector falsely regarded the release as completion of the mixing sub-task even when there were other materials that still needed to be mixed. Then, touching the other materials caused the system to recover from the error. This continued until all materials in the sub-task were mixed. In most cases, the failures described above were ignored by the subjects, and the algorithm recovered successfully in subsequent object accesses. In this sense, it was confirmed that our interface design and the error recovery mechanism worked well even for natural human activity.

Other types of failures occurred when objects were released in the middle of execution. For example, this situation was observed when a heating sub-task was executed in parallel with other sub-tasks and when the subjects took a short break.

All of the above failures were caused by failures to detect the completion of sub-tasks. These failures can bother users who react as subjects C and F did. To confirm the completion of each sub-task more accurately, we should employ other indicators besides object access. Addressing this problem will be a focus of future work.

5.3 Analysis of inputs from wizards

In previous sections, we have discussed the strategy of the proposed algorithm from a natural human activity perspective. As a matter of practice, the strategy also works robustly against negative effects caused by failures in the object sensor modules, as well as deceptive access and the use of unanticipated tools. We discuss this property in this subsection.

A user accesses objects frequently during a cooking task, and it is common to access three or four different objects in quick. Therefore, it is difficult even for wizards to operate the system without failures occurring. Such failures can be considered failures of the object sensor modules.

Table 5 shows the failures in the experiments for subjects A to E. These failures were confirmed after a careful survey of the video and operational logs. A false positive is a case in which a wizard inputs a touch for an untouched object and a release for an unreleased object. A false negative is a lack of input for touched and released objects. The precision and recall of input by a wizard were 94.4% and 73.5%, respectively. Generally, precision and recall demonstrate a trade-off relationship; however, the priority can be controlled by classifier parameters. In this sense, the wizards, which are the object sensor in the experiments, are a classifier controlled to yield higher precision and lower recall.

Table 6 shows a breakdown of the false positive inputs and the number of times false sub-tasks were displayed resulting from the false positive inputs. As can be seen, false positive inputs do not directly lead to displaying failures, and the algorithm has a certain level of robustness against failures in the object sensor module.

In addition to the analysis of errors in detection in object access, we checked errors in object recognition. Misrecognition shown in Table 5 represents an inappropriate switching of false positive and false negative. We considered that a false positive and a false negative is switched inappropriately when they occured within a maximum of 1 s. The number of misrecognition is small, and it is not possible to discuss the property of our method against misrecognition quantitatively. The system assumes human input

Table 5: Failures in wizard's inputs.

Subjects	A	B	C	D	E
# of false positive	15	9	2	10	9
# of collect input	176	199	143	164	163
# of total input	**191**	**208**	**145**	**174**	**172**
# of false negative (missing touch)	50	47	11	11	34
# of false negative (missing release)	52	44	11	11	33
# of misrecognition	1	2	0	1	1

Table 6: Comparison of number of false positive inputs and displaying false sub-tasks caused by false positive inputs.

Subjects		A		B		C		D		E	
Type	Actual Location	Input	Disp.	Input	Disp.	Input	Disp.	Input	Disp.	Input	Disp.
Touch	on the table	6	2	1	0	0	0	1	1	2	2
	in hand	1	0	3	0	1	0	4	3	3	0
Release	on the table	7	3	2	0	0	0	1	0	2	1
	in hand	0	1	3	0	1	0	4	2	2	0

during assistance to avoid cumulative error in progress tracking; thus, it is difficult to evaluate the system in an off-line simulation without modeling human behavior during the assistance. Therefore, quantitative analysis of misrecognition will be a focus of future work. In this paper, we only discuss the qualitative aspect of misrecognition errors.

In the experiment with subject A, an unforeseen "soup spoon" tool was misrecognized as a tablespoon. This led the system in a correct direction, and no failure was caused by this misrecognition. When subject B used the system, a rice paddle was switched to rice. This occurred due to the similarity of the related scenes. Both rice and rice paddle are related to similar sub-tasks, and this did not cause a failure in the displayed information. In the same trial, ginger was misrecognized as lettuce. This appears to be due to the arrangement of buttons in the wizard's interface. Different from the above misrecognition cases, this error disturbed the context and this error cannot led the system to a correct direction. Despite such out-of-context error, the system was not deceived. Misrecognitions in the trial with subjects D and E were both related to a label of mixture of materials, i.e., switching a mixture to a ladle, or starch to a mixture. These errors caused errors in display of sub-tasks. It seems that an access to mixture of materials indicates the cooking progress strondly; thus, misrecognition of a mixture of materials appears to affect more than misrecognition of a single material. Employing more sophisticated scoring rules than those shown in Table 1 can be a solution to improve robustness against these kind of errors. This also remains future work.

5.4 Discussion

A problem with maintaining V^- is identifying the *progress state* of the task. Schneider modeled progress transition using a DBN to simulate a cooking activity (Schneider, 2009). A set of cooking tasks was modeled by a randomly generated DBSs. In DBNs, each node corresponds to one progress state, and in their experiments, the number of nodes in DBN was 100, which is clearly too small to simulate a real cooking activity.

The progress state V^- can be coded as a binary assignment of a completion label to each $v \in V$, namely $label(v) = 1$ if $v \in V^-$ and vice versa. According to the method to calculate the number of possible progress states in G shown in Appendix B, there are 376 and 2940 different progress states in the ginger soup (Fig. 8) and fried rice (Fig. 9) graphs, respectively . Note that it is difficult to treat such a large number of states for a probabilistic graphical model, such as a DBN.

Rather than considering the probability in such a large-scale state space, we modeled the progress as a definite context V^+, and expanded it to V^b and V^f to deal with the uncertainty of human activity. This expansion strategy provided a good scope to accurately search and track the the progress state in the large-scale state space.

Besides the number of nodes, there are many edges that represent transitions from one state to another. In a real human activity, a worker select transitions along with many biases unconsciously. The objects in hand are a type of bias, and this bias is fully utilized in our method. Another bias is the structure of graph G. For example, in theory, the user can execute sub-task c in Fig. 2 after finishing sub-task a, but it is more likely that the user will execute sub-task b, which is the next sub-task on the same path as sub-task a. This bias is reflected in our system as the preliminarily defined order of the sub-tasks provided by R. Note that this order is not random; it holds some regularity along each path. This helps the system predict the forthcoming sub-task when there is no indication from object access. The proposed method worked successfully in real human activity because it considers these biases.

6 Conclusion

In this paper, we have proposed an intention-sensing interface for guiding cooking recipes. The system senses intention via the user accessing objects. Touching and releasing actions occur naturally in human activity; therefore, such actions can be a solution to assist humans performing sophisticated physical tasks organized by many sub-tasks in a workflow. The technical problems for utilizing object access are deceptive access and the use of unanticipated tools, both of which occur frequently in actual human activities. We have focused on such human factors, and developed an algorithm that is robust to deceptive access and unanticipated tools.

In experiments, we first compared the intention-sensing accuracy with two different recipes, i.e., ginger soup, which has unique object set O_v^i for any $v \in V$, and fried rice, which has some sub-tasks that share the same set of objects as O_v^i. To remove the human factors of deceptive accesses and unanticipated tools, we strongly restricted human activity in the first experiment. The former recipe was guided precisely by the system when there were no deceptive accesses or unforeseen tools. The latter recipe,

which was expected to be more difficult to guide, was also guided precisely in the first experiment. As a result, we can confirm that our algorithm has a certain level of robustness against recipe complexity while identifying the intended sub-tasks.

The second experiment was performed with natural human activity, i.e., we did not restrict subject activity. Rather, we laid out the situation to provoke unexpected activity. We used the fried rice recipe for this experiment because it is a more complex recipe. As a result, the sub-tasks were displayed with 73.6% accuracy on average for subjects A to E. From a human-computer interaction perspective, the number of operation by the subject is a direct criteria to evaluate the system performance. In the experiments, four subjects needed to input only two manual operations at most, and three, five, and nine times for subjects G, F, and C respectively, while guiding fried rice recipe that has 30 sub-tasks.

In this study, we designed the intention-sensing functionality based on accessing objects. Note that there are several options to enhance this functionality. First, the location and posture of the hand (Song et al., 2013) and those of objects are important for human intention estimation. This should eliminate deceptive accesses and improve the accuracy of the proposed system. Second, it will allow the system to predict user intension earlier and possibly more accurately to replace touch detection by touch forecasting through gaze estimation (Nakazawa & Nitschke, 2012), grabbing motion estimation (Prasad, Kellokumpu, & Davis, 2006), or observing activity with a wrist-mounted camera (Ohnishi, Kanehira, Kanezaki, & Harada, 2015). Testing the system with such options remains a focus of future work.

There are many additional topics to be studied because our system is a pilot system. First, we must develop an object sensor for different tasks. The writing cost of recipes in the workflow graph representation is another topic. Some studies have attempted to extract the structure of sub-tasks automatically from traditional text recipes (Mori, Maeta, Yamakata, & Sasada, 2014). In addition, the recommendation function should be further investigated to obtain higher usability. Last, to detect the completion of sub-tasks at a practical cost, we must consider other effective indicators in addition to object access. All of these tasks remain future work.

A Algorithm to assign materials to each node in workflow

Let $G(V, E)$ be a digraph representing a recipe workflow. Consider a sub-task v as a process that receives one or more materials as input and outputs one material. The input materials correspond to O_v^m, and o_v^m denotes the output material from $v \in V$.

When v is a leaf node, we manually assign O_v^m (A to a and B to c in Fig. 2). Otherwise, O_v^m is obtained as a set of outputs from nodes that have an out-edge to v.

Let u_i be a node that is adjacent to v by v's i-th in-edge $\{u_i, v\} \in E$. When v has only one in-edge (thus, i is always 0), O_v^m consists of only one unique element. We use the same label for a material as long as they are not mixed with other materials; thus, o_v^m is equal to the unique element in O_v^m. When v has several in-edges (d in Fig. 2), O_v^m contains several materials and o_v^m is obtained as the mixture of those materials, i.e.,

$o_v^m = O_v^m = \cup_{u_i} o_{u_i}^m$ (at d in Fig. 2, $O_d^m = \{A, B\}$ and it becomes the unique input element at e, i.e., $\{\{A, B\}\} = O_e^m$).

This definition is compatible with systems that can track materials over sub-tasks (Hashimoto et al., 2010), or those driven by RFID in an assembly context.

B Number of possible progress states in a flexible recipe

Let $G(V, E)$ be a directed graph representing the workflow of a recipe, where $v \in V$ corresponds to a sub-task and $\{u, v\} \in E$ is a directed edge. A sub-task u must be completed to begin v if $\{u, v\} \in E$.

To count the possible progress states (PPSs), we consider a local progress state around v;

$$N_v = \{u \mid \{u, v\} \in E\} \tag{4}$$

$$S(v) = \begin{cases} \{l(u) \mid u \in N_v\} & N_v \neq \phi \\ \phi & N_v = \phi, \end{cases} \tag{5}$$

where $l(u) = \{0, 1\}$ is a Boolean variable indicating the completion of sub-task u ($l(u) = 1$ for u's completion). v is ready to be executed if all Boolean variables in $S(v)$ are true (or $S(v) = \phi$).

For simplicity, we assume that G is a tree. This assumption is true for most recipes. We note that the following estimation will not decrease drastically even if there are a few acyclic closed paths in G. Let us consider subtree G_v cut from G by node v, i.e., a subgraph induced from v and all ancestor nodes of v. Then, the number of PPSs in subtree G_v is counted by function $\#(S(v))$ as

$$\#(S(v)) = \begin{cases} 1 + \Pi_{u \in N_v} \#(S(u)) & N_v \neq \phi \\ 2 & N_v = \phi, \end{cases} \tag{6}$$

where 1 is added to count the following two cases differently, i.e., the cases where v is ready but not completed, and where v has been completed. The same cases are also counted when $N_v = \phi$.

The total number of PPSs in G is obtained as $\#(S(v^t))$, where v^t is the sink node of workflow graph G. Due to the direct product property of Eq. (6), $\#(S)$ increases combinatorially at each conjunction on the workflow G; thus, it cannot be treated by general DBN models.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers 24240030, 26280084 and 16K16099.

References

Bradbury, J. S., Shell, J. S., & Knowles, C. B. (2003). Hands on cooking: Towards an attentive kitchen. In *Proc. of chi '03 extended abstracts on human factors in computing systems* (pp. 996–997).

Chang, K.-h., Liu, S.-y., Chu, H.-h., Hsu, J. Y.-j., Chen, C., Lin, T.-Y., ... Huang, P. (2006). The diet-aware dining table: Observing dietary behaviors over a tabletop surface. In *Proc. of the 4th international conference on pervasive computing* (pp. 366–382).

Favela, J., Tentori, M., & Gonzalez, V. M. (2010). Ecological validity and pervasiveness in the evaluation of ubiquitous computing technologies for health care. *Intl. Journal of Human–Computer Interaction*, *26*(5), 414–444.

Hamada, R., Okabe, J., Ide, I., Sakai, S., & Tanaka, H. (2005). Cooking navi: Assistant for Daily Cooking in Kitchen. In *Proc. of the 13th annual acm international conference on multimedia* (p. 371-374). ACM Press New York, USA.

Hashimoto, A., Inoue, J., Nakamura, K., Funatomi, T., Ueda, M., Yamakata, Y., & Minoh, M. (2012). Recognizing ingredients at cutting process by integrating multimodal features. In *Proc. of the ACM multimedia 2012 workshop on multimedia for cooking and eating activities* (pp. 13–18).

Hashimoto, A., Mori, N., Funatomi, T., Mukunoki, M., Kakusho, K., & Minoh, M. (2010). Tracking food materials with changing their appearance in food preparing. In *Proc. of ISM 2010 workshop on multimedia for cooking and eating activities* (pp. 248–253).

Iscen, A., & Duygulu, P. (2013). Knives are picked before slices are cut: Recognition through activity sequence analysis. In *Proc. of the 5th international workshop on multimedia for cooking and eating activities* (pp. 3–8).

Ju, W., Hurwitz, R., Judd, T., & Lee, B. (2001). Counteractive: an interactive cookbook for the kitchen counter. In *Proc. of chi '01 extended abstracts on human factors in computing systems* (pp. 269–270). ACM Press New York, USA.

Kelley, J. F. (1984). An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems*, *2*(1), 26–41.

Klompmaker, F., Nebe, K., & Fast, A. (2012). dSensingNI a framework for advanced tangible interaction using a depth camera. In *Proc. of the sixth international conference on tangible, embedded and embodied interaction* (pp. 217–224).

Kuehne, H., Fraunhofer, F., Arslan, A., & Serre, T. (2014). The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proc. of 2014 IEEE conference on computer vision and pattern recognition* (p. 780 - 787).

Lei, J., Ren, X., & Fox, D. (2012). Fine-grained kitchen activity recognition using RGB-D. In *Proc. of the 2012 acm conference on ubiquitous computing* (pp. 208–211).

López, G., López, M., Guerrero, L. A., & Bravo, J. (2014). Human–objects interaction: A framework for designing, developing and evaluating augmented objects. *International Journal of Human-Computer Interaction*, *30*(10), 787–801.

Matsushima, Y., Funabiki, N., Zhang, Y., Nakanishi, T., & Watanabe, K. (2013). Extensions of cooking guidance function on android tablet for homemade cooking assistance system. In *IEEE 2nd global conference on consumer electronics* (pp. 397–401).

Miyawaki, K., & Sano, M. (2008). A virtual agent for a cooking navigation system using augmented reality. In *Proc. of 8th international conference on intelligent virtual agents* (pp. 97–103).

Mori, S., Maeta, H., Yamakata, Y., & Sasada, T. (2014). Flow graph corpus from recipe texts. In *Proc. of the nineth international conference on language resources and evaluation.*

Murata, M. (2001). *RELAX NG Home Page.* `http://relaxng.org/`.

Nakauchi, Y., Fukuda, T., Noguchi, K., & Matsubara, T. (2005). Intelligent Kitchen: Cooking Support by LCD and Mobile Robot with IC-Labeled Objects. In *Proc. of ieee/rsj international conference on intelligent robots and systems* (pp. 1911–1916).

Nakazawa, A., & Nitschke, C. (2012). Point of gaze estimation through corneal surface reflection in an active illumination environment. In *Proc. of european conference on computer vision* (p. 159-172).

Nintendo. (2006). *Syaberu! DS Oryouri Navi.* http://www.nintendo.co.jp/ds/a4vj/.

Ohnishi, K., Kanehira, A., Kanezaki, A., & Harada, T. (2015). Recognizing activities of daily living with a wrist-mounted camera. *CoRR*, *abs/1511.06783*. Retrieved from `http://arxiv.org/abs/1511.06783`

Packer, B., Saenko, K., & Koller, D. (2012). A combined pose, object, and feature model for action understanding. In *Proc. of 2012 IEEE conference on computer vision and pattern recognition* (pp. 1378–1385).

Prasad, V. S. N., Kellokumpu, V., & Davis, L. S. (2006). Ballistic hand movements. In *Proc. of articulated motion and deformable objects* (pp. 153–164).

Rohrbach, M., Amin, S., Andriluka, M., & Schiele, B. (2012). A database for fine grained activity eetection of cooking activities. In *Proc. of 2012 IEEE conference on computer vision and pattern recognition* (pp. 1194–1201).

Sandweg, N., Hassenzahl, M., & Kuhn, K. (2000). Designing a telephone-based interface for a home automation system. *International Journal of Human-Computer Interaction*, *12*(3-4), 401–414.

Schneider, M. (2009). Plan recognition in instrumented environments. In *Proc. of the 5th international conference on intelligent environments* (pp. 295–302).

Shimada, A., Kondo, K., Deguchi, D., Morin, G., & Stern, H. (2013). Kitchen scene context based gesture recognition: A contest in icpr2012. In *Advances in depth image analysis and applications* (pp. 168–185). Springer.

Siltanen, S., Hakkarainen, M., Korkalo, O., Salonen, T., Saaski, J., Woodward, C., ... Potamianos, A. (2007). Multimodal user interface for augmented assembly. In *Proc. of IEEE 9th workshop on multimedia signal processing* (pp. 78–81).

Song, D., Kyriazis, N., Oikonomidis, I., Papazov, C., Argyros, A., Burschka, D., & Kragic, D. (2013). Predicting human intention in visual observations of hand/object interactions. In *Proc. of 2013 ieee international conference on robotics and automation,* (pp. 1608–1615).

Tang, A., Owen, C., Biocca, F., & Mou, W. (2003). Comparative effectiveness of augmented reality in object assembly. In *Proc. of the SIGCHI conference on human factors in computing systems* (pp. 73–80).

Ueda, M., Funatomi, T., Hashimoto, A., Watanabe, T., & Minoh, M. (2011). Developing a real-time system for measuring the consumption of seasoning. In *Proc. of ieee ISM 2011 workshop on multimedia for cooking and eating activities* (pp. 393–398).

Uriu, D., Namai, M., Tokuhisa, S., Kashiwagi, R., Inami, M., & Okude, N. (2012). panavi: Recipe medium with a sensors-embedded pan for domestic users to master professional culinary arts. In *Proc. of the SIGCHI conference on human factors in computing systems* (pp. 129–138).

Yasuoka, R., Hashimoto, A., Funatomi, T., & Minoh, M. (2013). Detecting start and end times of object-handlings on a table by fusion of camera and load sensors. In *Proc. of the 5th international workshop on multimedia for cooking and eating activities* (pp. 51–56).

Yuan, M. L., Ong, S. K., & Nee, A. Y. C. (2008). Augmented reality for assembly guidance using a virtual interactive tool. *International Journal of Production Research*, *46*(7), 1745-1767.

Zauner, J., Haller, M., Brandl, A., & Hartmann, W. (2003). Authoring of a mixed reality assembly instructor for hierarchical structures. In *Proc. of the second IEEE and ACM international symposium on mixed and augmented reality* (pp. 237–246).

Authors' Biographies

Atsushi Hashimoto is an Assistant Professor at the Graduate School of Education, Kyoto University, Japan. He received B.S. degree in Engineering, and M.S. and Ph.D. degrees in Informatics from Kyoto University, in 2005, 2008, and 2013, respectively. His research interests include pattern recognition, computer vision, and human-computer interaction.

Jin Inoue received B.S. degree in Engineering, and M.S. degree in Informatics from Kyoto University, Japan, in 2011 and 2013, respectively.

Takuya Funatomi is an Associate Professor at Information Science Department, Nara Institute of Science and Technology, Japan. He received M.S. and Ph.D. degrees in Informatics from Kyoto University, in 2004 and 2007, respectively. He is a member of ACM, and the IEEE Computer Society and Communication Society.

Michihiko Minoh is a Professor at the Academic Center for Computing and Media Studies, Kyoto University, Japan. He received BEng, MEng, and DEng degrees in Information Science from Kyoto University in 1978, 1980, and 1983, respectively. He is a member of the IEEE Computer Society and Communication Society, and ACM.